009390 * 20950960

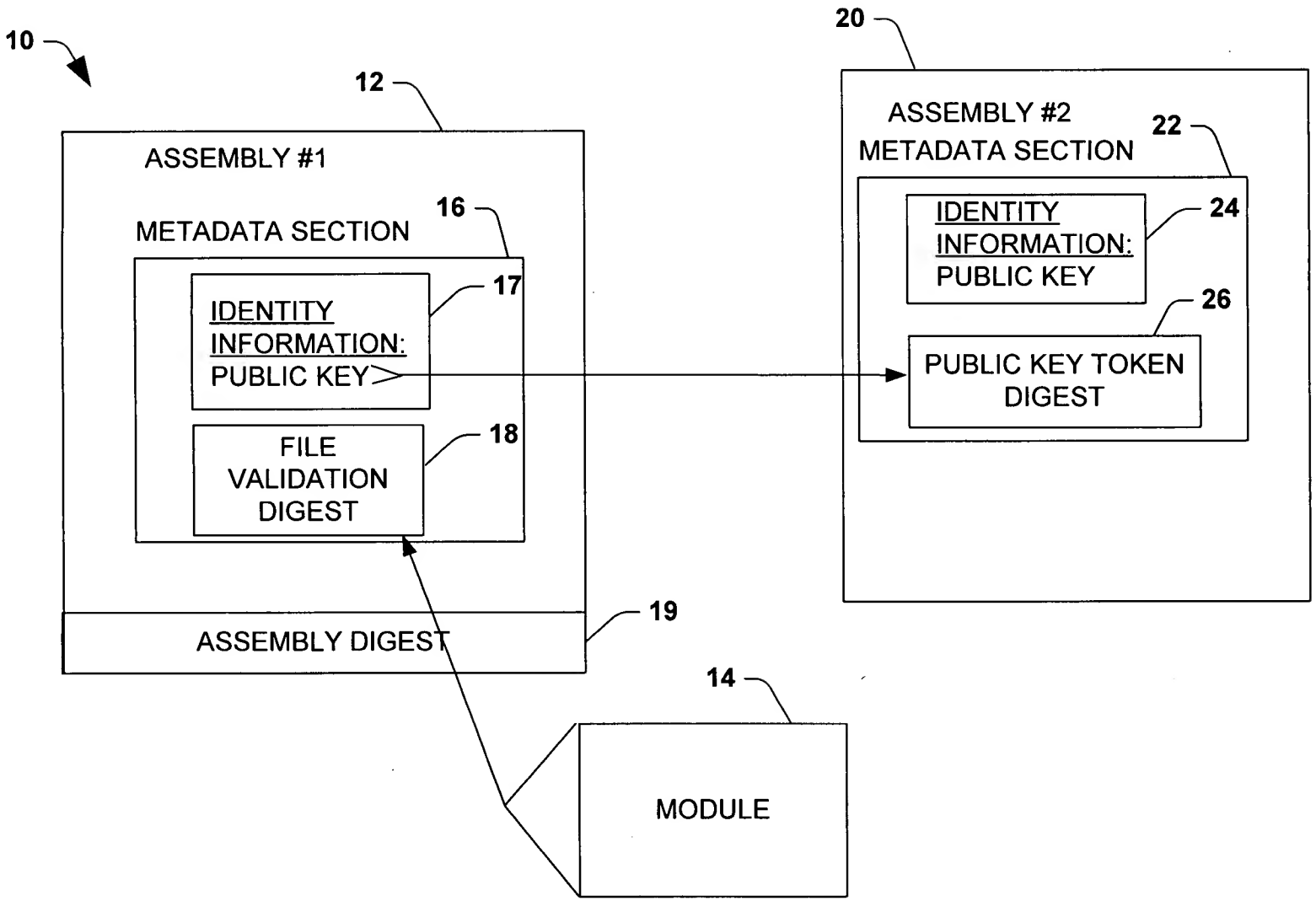

Fig. 1

09/605,602

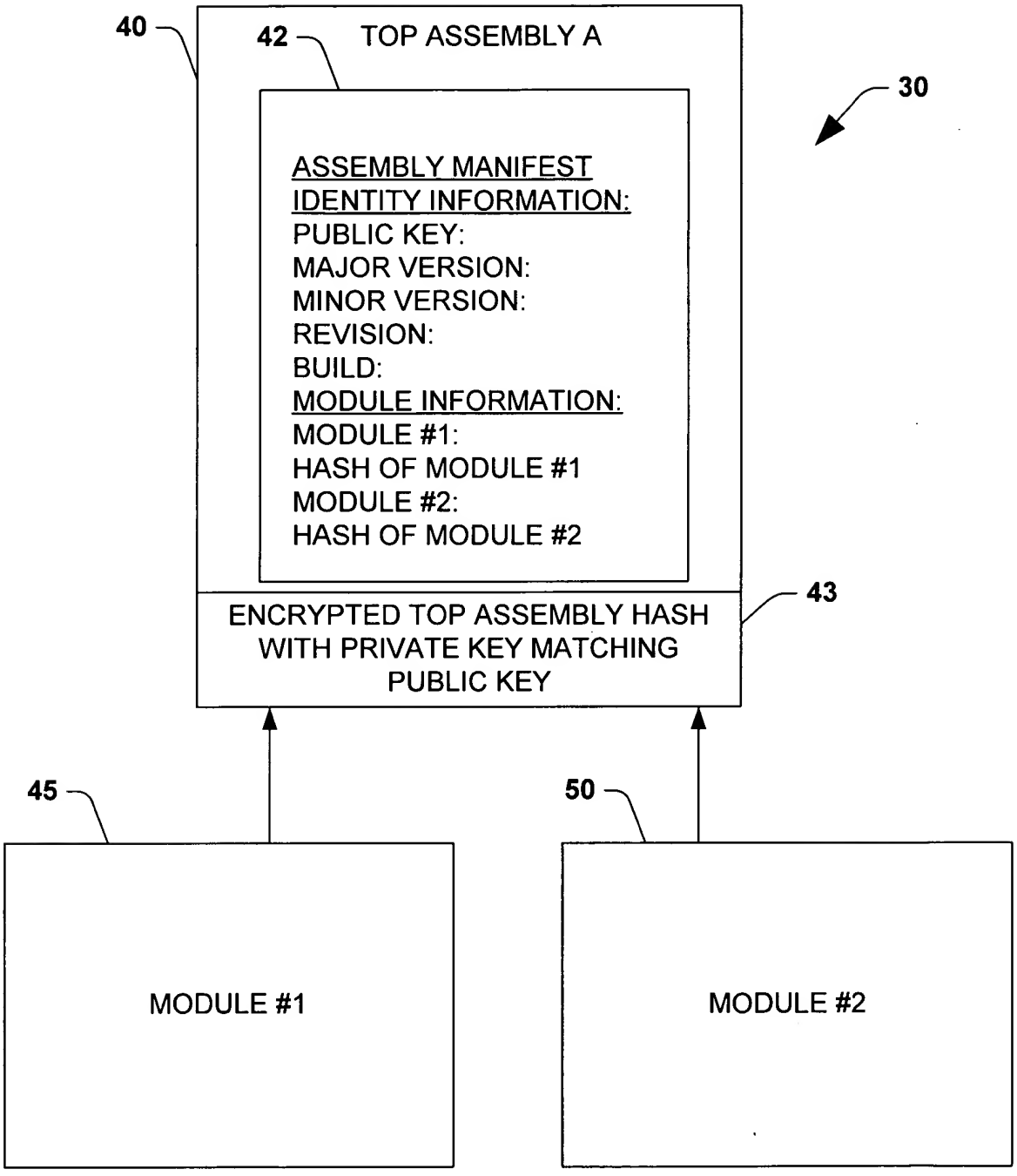
40
42
TOP ASSEMBLY A
30
ASSEMBLY MANIFEST
IDENTITY INFORMATION:
PUBLIC KEY:
MAJOR VERSION:
MINOR VERSION:
REVISION:
BUILD:
MODULE INFORMATION:
MODULE #1:
HASH OF MODULE #1
MODULE #2:
HASH OF MODULE #2
43
ENCRYPTED TOP ASSEMBLY HASH
WITH PRIVATE KEY MATCHING
PUBLIC KEY
45
50
MODULE #1
MODULE #2

Fig. 2

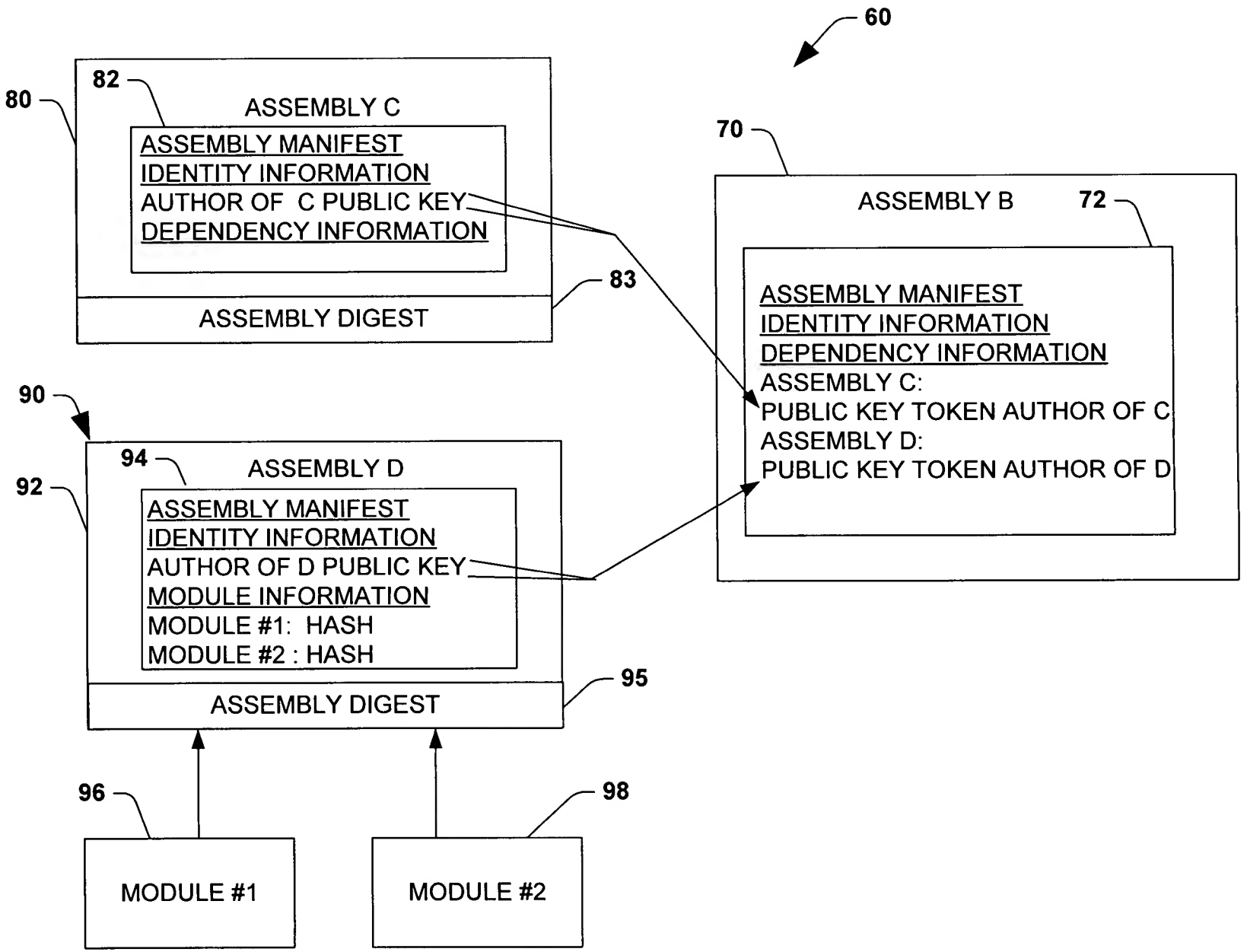
60
80
82
ASSEMBLY C
ASSEMBLY MANIFEST
IDENTITY INFORMATION
AUTHOR OF C PUBLIC KEY
DEPENDENCY INFORMATION
ASSEMBLY DIGEST
83
70
ASSEMBLY B
72
ASSEMBLY MANIFEST
IDENTITY INFORMATION
DEPENDENCY INFORMATION
ASSEMBLY C:
PUBLIC KEY TOKEN AUTHOR OF C
ASSEMBLY D:
PUBLIC KEY TOKEN AUTHOR OF D
90
92
94
ASSEMBLY D
ASSEMBLY MANIFEST
IDENTITY INFORMATION
AUTHOR OF D PUBLIC KEY
MODULE INFORMATION
MODULE #1: HASH
MODULE #2 : HASH
ASSEMBLY DIGEST
95
96
98
MODULE #1
MODULE #2

Fig. 3

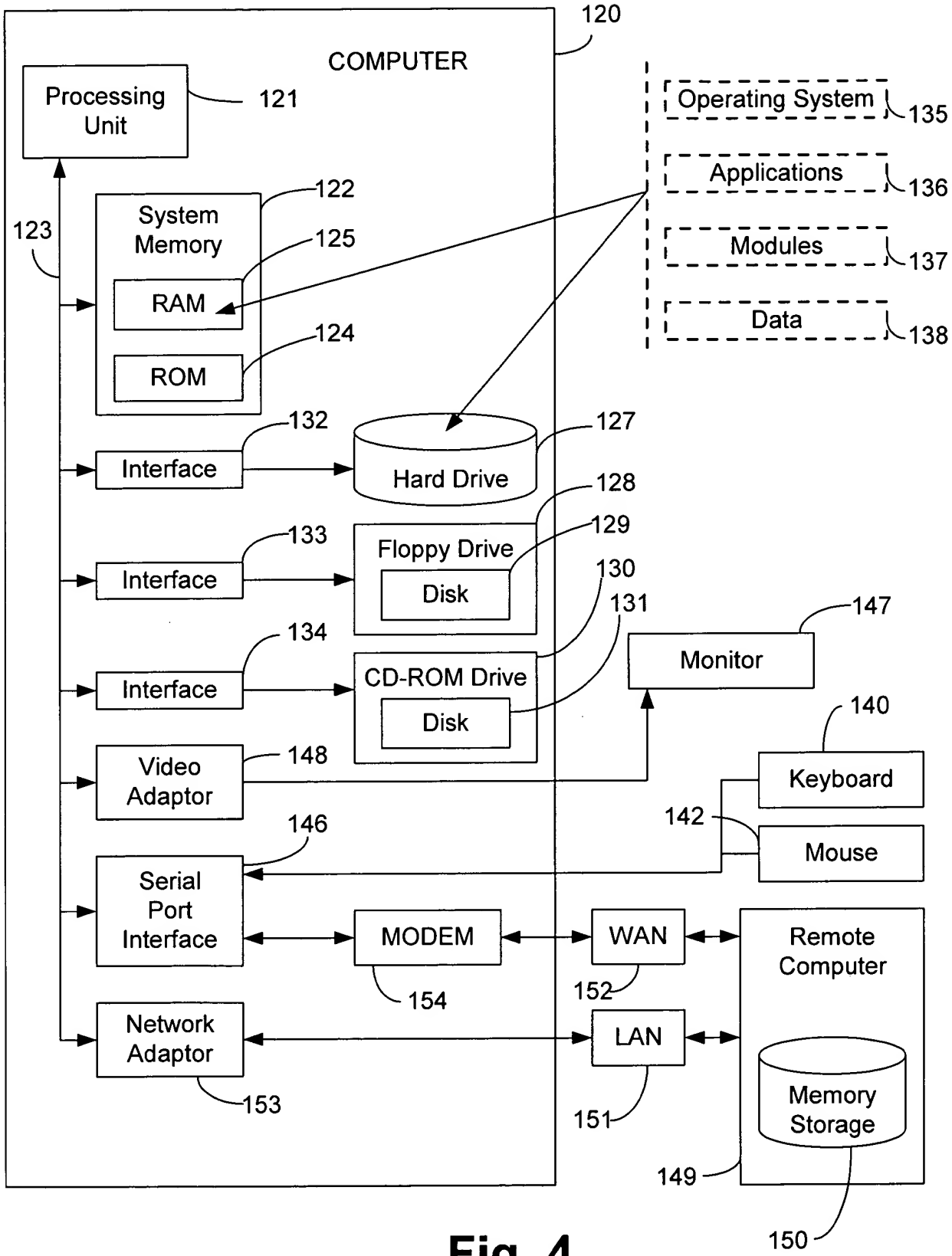
120
COMPUTER
Processing Unit
121
Operating System
135
Applications
136
Modules
137
Data
138
System Memory
122
123
125
RAM
124
ROM
Interface
132
Hard Drive
127
128
Floppy Drive
129
Interface
133
Disk
130
131
CD-ROM Drive
Interface
134
Disk
147
Monitor
140
Video Adaptor
148
Keyboard
142
Mouse
146
Serial Port Interface
MODEM
WAN
Remote Computer
154
152
Network Adaptor
LAN
Memory Storage
153
151
149
150

Fig. 4

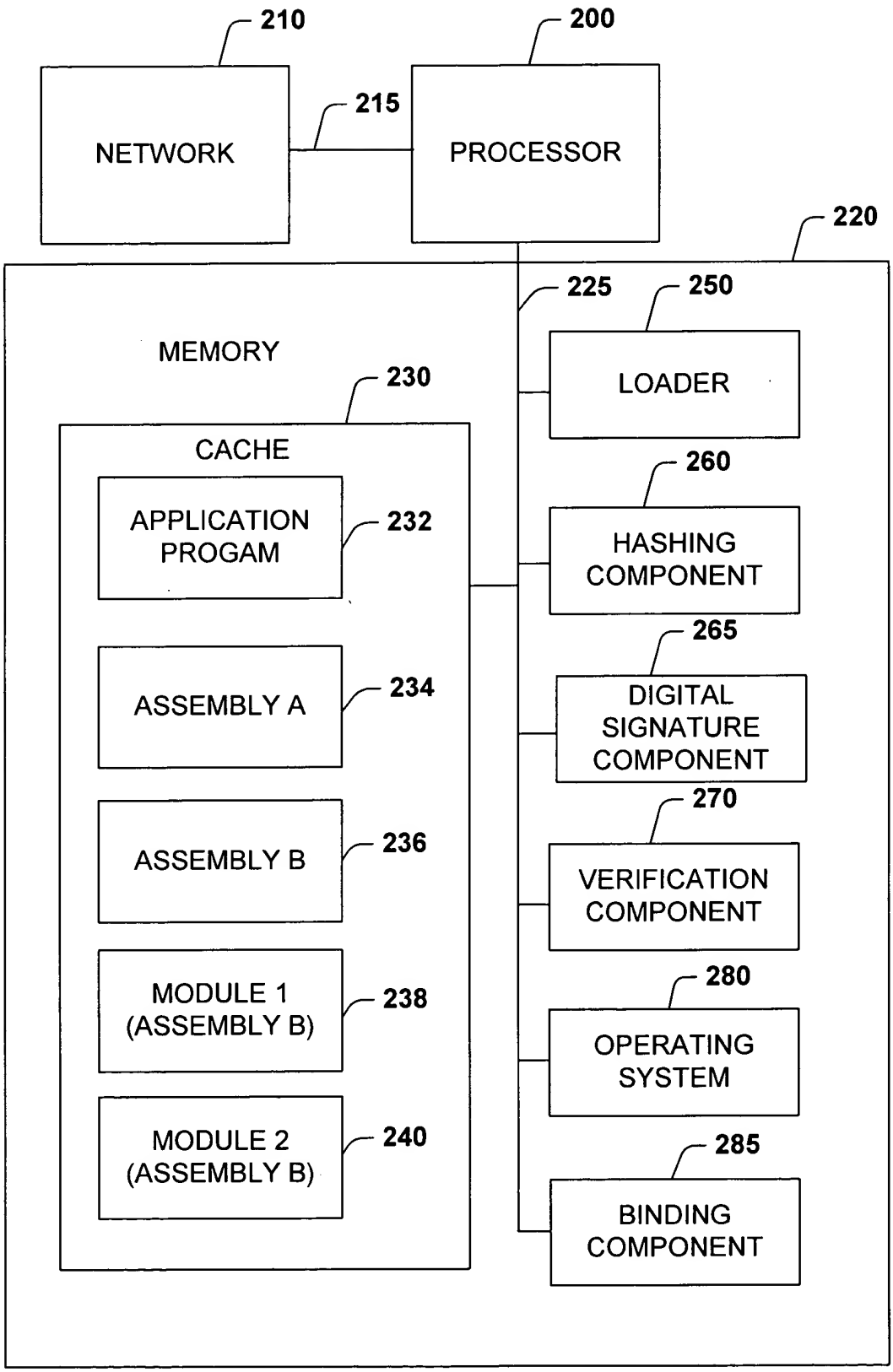
210
200
NETWORK
215
PROCESSOR
220
225
250
MEMORY
LOADER
230
CACHE
260
APPLICATION PROGAM
232
HASHING COMPONENT
265
ASSEMBLY A
234
DIGITAL SIGNATURE COMPONENT
270
ASSEMBLY B
236
VERIFICATION COMPONENT
MODULE 1 (ASSEMBLY B)
238
280
OPERATING SYSTEM
MODULE 2 (ASSEMBLY B)
240
285
BINDING COMPONENT

Fig. 5

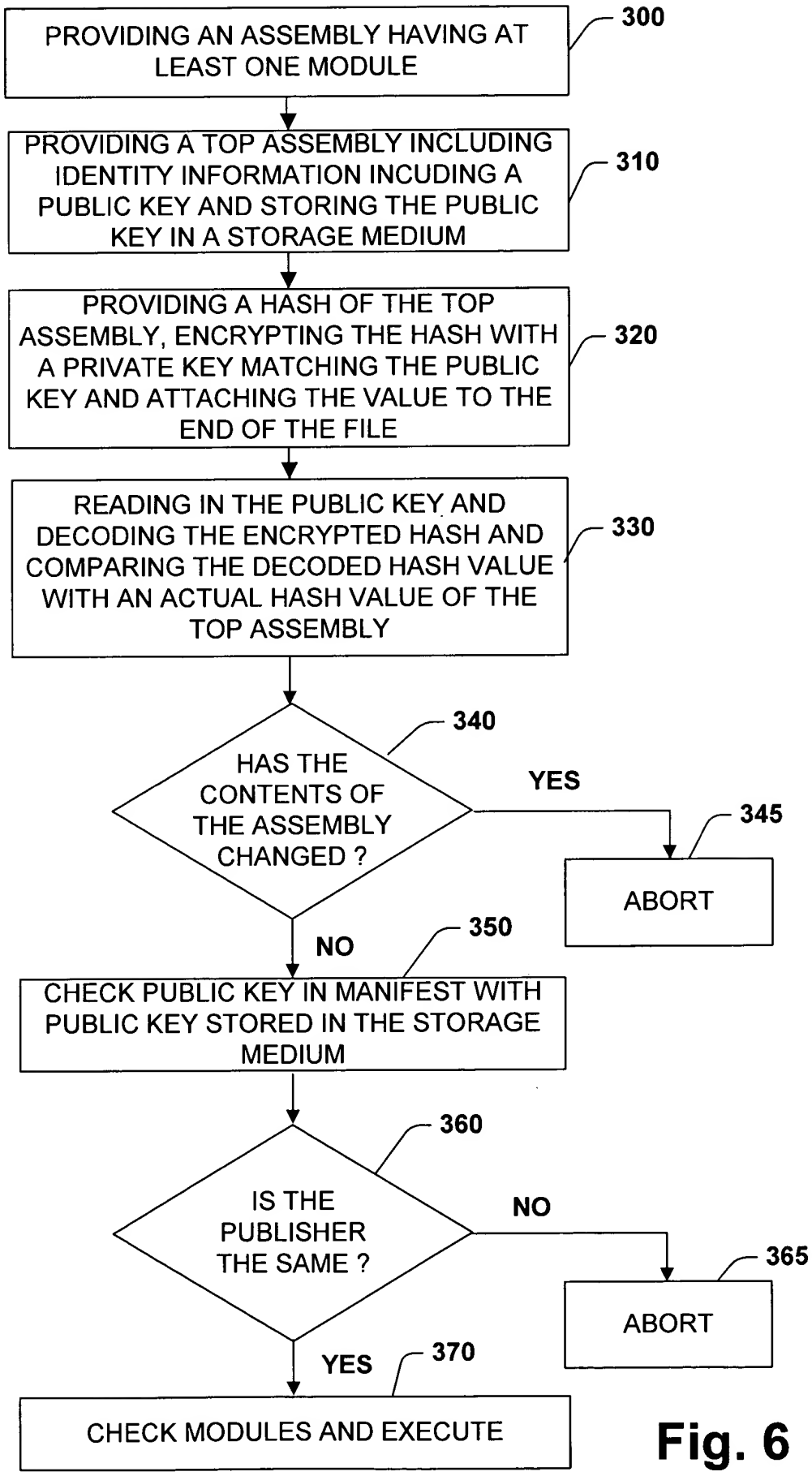
PROVIDING AN ASSEMBLY HAVING AT LEAST ONE MODULE
300
PROVIDING A TOP ASSEMBLY INCLUDING IDENTITY INFORMATION INCUDING A PUBLIC KEY AND STORING THE PUBLIC KEY IN A STORAGE MEDIUM
310
PROVIDING A HASH OF THE TOP ASSEMBLY, ENCRYPTING THE HASH WITH A PRIVATE KEY MATCHING THE PUBLIC KEY AND ATTACHING THE VALUE TO THE END OF THE FILE
320
READING IN THE PUBLIC KEY AND DECODING THE ENCRYPTED HASH AND COMPARING THE DECODED HASH VALUE WITH AN ACTUAL HASH VALUE OF THE TOP ASSEMBLY
330
HAS THE CONTENTS OF THE ASSEMBLY CHANGED ?
340
YES
345
ABORT
NO
350
CHECK PUBLIC KEY IN MANIFEST WITH PUBLIC KEY STORED IN THE STORAGE MEDIUM
IS THE PUBLISHER THE SAME ?
360
NO
365
ABORT
YES
370
CHECK MODULES AND EXECUTE

Fig. 6

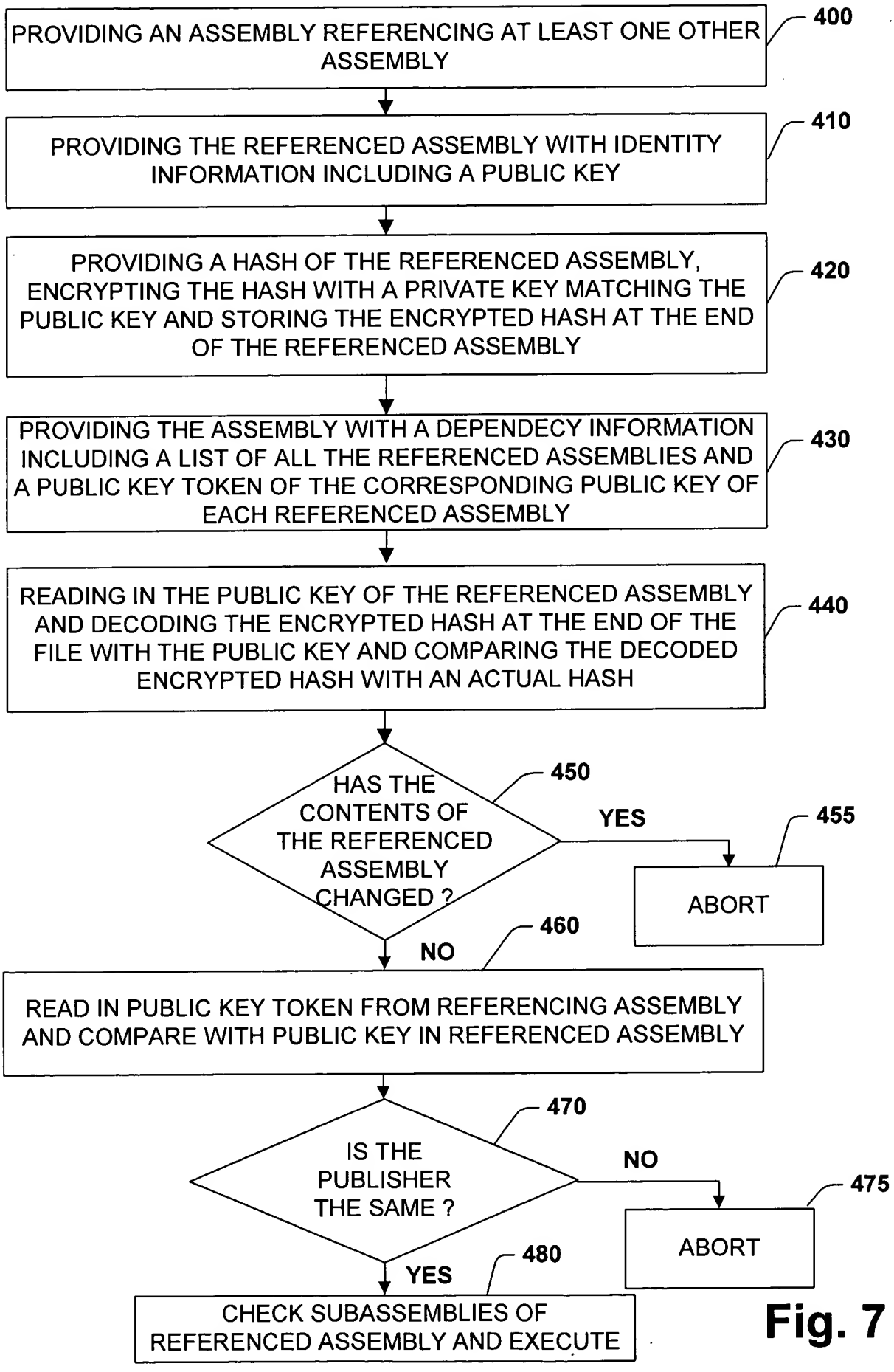
PROVIDING AN ASSEMBLY REFERENCING AT LEAST ONE OTHER ASSEMBLY
400
PROVIDING THE REFERENCED ASSEMBLY WITH IDENTITY INFORMATION INCLUDING A PUBLIC KEY
410
PROVIDING A HASH OF THE REFERENCED ASSEMBLY, ENCRYPTING THE HASH WITH A PRIVATE KEY MATCHING THE PUBLIC KEY AND STORING THE ENCRYPTED HASH AT THE END OF THE REFERENCED ASSEMBLY
420
PROVIDING THE ASSEMBLY WITH A DEPENDECY INFORMATION INCLUDING A LIST OF ALL THE REFERENCED ASSEMBLIES AND A PUBLIC KEY TOKEN OF THE CORRESPONDING PUBLIC KEY OF EACH REFERENCED ASSEMBLY
430
READING IN THE PUBLIC KEY OF THE REFERENCED ASSEMBLY AND DECODING THE ENCRYPTED HASH AT THE END OF THE FILE WITH THE PUBLIC KEY AND COMPARING THE DECODED ENCRYPTED HASH WITH AN ACTUAL HASH
440
HAS THE CONTENTS OF THE REFERENCED ASSEMBLY CHANGED ?
450
YES
455
ABORT
NO
460
READ IN PUBLIC KEY TOKEN FROM REFERENCING ASSEMBLY AND COMPARE WITH PUBLIC KEY IN REFERENCED ASSEMBLY
IS THE PUBLISHER THE SAME ?
470
NO
475
ABORT
YES
480
CHECK SUBASSEMBLIES OF REFERENCED ASSEMBLY AND EXECUTE

Fig. 7

```
BOOL SharedNameKeyGen(
      // [in] desired key container name, must be a non-
empty string
      LPCWSTR          szKeyContainer,
      // [in] flag parameter, leave key installed is only
option
      DWORD             dwFlags,
      // [out] public/private key blob
      BYTE             **ppbKeyBlob,
      ULONG             *pcbKeyBlob)
```

Fig. 8a

```
BOOL SharedNameKeyInstall(
      // [in] desired key container name, must be a non-empty
      //string
      LPCWSTR         szKeyContainer,
      // [in] public/private key blob
      BYTE             *pbKeyBlob,
      ULONG             cbKeyBlob)
```

Fig. 8b

```
BOOL SharedNameKeyDelete(
      // [in] desired key container name
      LPCWSTR          szContainer)
```

Fig. 8c

```
VOID SharedNameFreeBuffer(
   // [in] pointer to memory to be freed
 BYTE      *pbMemory)
```

Fig. 8d

```
BOOL SharedNameGetPublicKey (
      // [in] desired key container name
      LPCWSTR          szContainer,
      // [in] public/private key blob (optional)
      BYTE             *pbKeyBlob,
      ULONG             cbKeyBlob
      // [out] public key blob
      BYTE             **ppbPublicKeyBlob,
      ULONG             *pcbPublicKeyBlob)
```

Fig. 9a

```
Struct {
      ALG_ID   SigAlgID;      // strong name signature algorithm
      ALG_ID   HashAlgID;     // strong name hash algorithm
      ULONG      cbPublicKey; // length of the key in bytes
      BTYE[]   PublicKey;     // variable length byte array
                               containing the key value in format
                               output by CryptoAPI
   } PublicKeyBlob;
```

Fig. 9b

```
BOOL SharedNameSignatureGeneration(
      // [in] valid path to the PE file for the Assembly
      LPCWSTR          szFilePath,
      // [in] name of key, if NULL will default to SharedNameKey
      LPCWSTR        szKeyContainer,
      // [in] public/private key blob (optional)
      BYTE             *pbKeyBlob,
      ULONG             cbKeyBlob
   // [in/out] signature blob
      BYTE              **ppbSignatureBlob,
      ULONG           *pcbSignatureBlob)
```

Fig. 9c

```
BOOL SharedNameTokenFromAssembly(
    // [in] valid path to the PE file for the Assembly
    LPCWSTR          szFilePath,
    // [out] strong name token
    BYTE              **ppbSharedNameToken,
ULONG      *pcbSharedNameToken)
```

Fig. 10a

```
BOOL SharedNameTokenFromPublicKey(
      // [in] public key blob
      BYTE             *pbPublicKeyBlob,
      ULONG             cbPublicKeyBlob)
      // [out] strong name token
      BYTE               **ppbSharedNameToken,
      ULONG            *pcbSharedNameToken)
```

Fig. 10b

```
BOOL SharedNameSignatureVerification(
      // [in] valid path to the PE file for the Assembly
      LPCWSTR               szFilePath)
```

Fig. 10c